PACKAGENT: EVOLVING THREE COORDINATED SKILLS FOR STATE-ADAPTIVE 3D BIN PACKING

Anonymous authors
Paper under double-blind review

ABSTRACT

The three-dimensional bin packing problem aims to pack heterogeneous items into as few bins as possible while satisfying geometric and physical constraints, including containment, non-overlap, and support. Item selection and placement decisions must adapt to changes in the remaining items and the current spatial configuration, whereas fixed heuristics struggle to accommodate the decision requirements of different packing phases. To address this challenge, we propose PackAgent, a three-skill evolutionary framework for constrained multi-bin three-dimensional packing. PackAgent decomposes the core decisions into three skills: state reflection ε, item selection ϕ, and placement scoring ψ. A large language model jointly evolves these executable skills offline. At each placement step, ε generates phase cues and factor weights from the current packing state, dynamically modulating the decision preferences of ϕ and ψ. Candidate generation, rotation enumeration, and constraint verification are handled by a fixed solver to ensure geometric feasibility. Unlike automated heuristic design methods that search for a single fixed rule or rely on post-hoc selection among multiple strategies, PackAgent enables state-adaptive decisions through step-level factor modulation within a single packing trajectory. We evaluate PackAgent on the SSSCSP benchmark under offline, online, and semi-online settings, as well as across instance scales on the IVC dataset. PackAgent achieves the lowest average bin count among the compared single-trajectory methods in all three information-visibility settings. Deployment ablations further demonstrate the importance of dynamic modulation through state reflection for coordinating item selection and placement scoring. Code is provided as anonymized supplementary material.

1 INTRODUCTION

The three-dimensional bin packing problem (3D-BPP) arises widely in warehousing, logistics, and container transportation. Its objective is to pack heterogeneous items into as few bins as possible while satisfying geometric and physical constraints. Unlike lower-dimensional packing problems, solution quality in 3D-BPP depends jointly on the item loading order, rotation orientations, and three-dimensional placement positions. Each local placement reshapes the remaining free space and may reduce the support surfaces available to subsequent items Goyal & Deng (2020); Kar et al. (2025). The resulting spatial fragmentation, wasted vertical space, and support conflicts accumulate along the packing sequence. High-quality decisions therefore depend on both the current local configuration and the remaining items and visible future information.

Existing approaches to three-dimensional bin packing mainly include manually designed heuristics, search-based optimization methods, and deep reinforcement learning (DRL). Combinations of manually designed heuristics and search-based optimization offer low inference overhead and can readily incorporate deterministic constraint checks Santos et al. (2019); Gzara et al. (2020). However, their rules typically rely on expert knowledge and are highly task-specific, limiting their adaptability to changes in item attributes, pallet dimensions, and operational constraints. DRL methods learn complex mappings from states to actions and have been applied to three-dimensional packing tasks with various constraints Pan et al. (2023); Zhao et al. (2026). Nevertheless, their training, adaptation, and deployment typically depend on specific model architectures and data distributions. Changes in item distributions or pallet dimensions often require retraining or fine-tuning, while policy decisions may lack sufficient interpretability. These limitations motivate an approach that preserves

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

interpretable decision logic, reduces the need for manually designed heuristic rules, and flexibly adapts to different packing scenarios and states.

Recent advances in large language model (LLM)-driven automated heuristic design (AHD) provide a new way to reduce manual design effort Pourcel et al. (2025); Zhao & Cheong (2026). Existing approaches typically search for a single executable heuristic by optimizing its average performance on training instances. EoH co-evolves natural-language descriptions and heuristic code, while ReEvo uses performance comparisons between candidates to guide reflective evolution Romera-Paredes et al. (2024); Liu et al. (2024); Ye et al. (2024). These approaches ultimately seek a heuristic that performs well across an entire solution trajectory. However, when its decision factors and weights remain fixed throughout packing, the resulting heuristic has limited ability to accommodate the requirements of different packing phases. EoH-S further introduces automated heuristic set design, searching for complementary heuristics to improve coverage across instances Liu et al. (2026). Its performance gains, however, rely on executing multiple members on the same task and selecting among their completed solutions, which incurs additional computational overhead. This motivates an evolutionary approach that executes only one member while dynamically adjusting its decision factors and weights at each step according to the current state.

To this end, we propose PackAgent, a three-skill evolutionary framework for constrained multi-bin three-dimensional packing. Instead of assigning the entire packing process to a single fixed heuristic or several independent heuristics, PackAgent decomposes the core decisions of each placement step into three evolvable skills: state reflection ε, item selection ϕ, and placement scoring ψ. At each packing step, ε reads the current state and dynamically determines the weights of the heuristic factors used by ϕ and ψ. It does not directly select an item or a candidate placement; instead, it provides a step-level control signal for subsequent decisions. Conditioned on the currently observable items and this control signal, ϕ selects the next item to pack. Meanwhile, ψ scores candidate three-dimensional placements that have passed constraint verification and determines the final placement. PackAgent confines state-dependent adaptation to these three interpretable skills, enabling the same agent to adjust its decision preferences at each step under offline, online, and semi-online settings.

Our main contributions are as follows:

1. We propose PackAgent, a three-skill evolutionary framework for constrained multi-bin three-dimensional packing. The framework formalizes the core packing decisions under offline, online, and semi-online settings and organizes them into three evolvable skills: state reflection ε, item selection ϕ, and placement scoring ψ. By jointly evolving these skills, an LLM can optimize the principal packing decisions within a fixed candidate-generation and constraint-verification framework.
2. We introduce a dynamic weighting mechanism for heuristic factors. At each packing step, the state reflection skill ε adjusts the factor weights used by ϕ and ψ according to the current state. This enables a single agent to adapt to different packing phases without executing multiple heuristics and selecting among their completed solutions.
3. We evaluate the framework on the SSSCSP benchmark under offline, online, and semi-online settings, as well as across instance scales on the IVC dataset. PackAgent achieves the lowest average bin count among the compared single-trajectory methods, and deployment ablations further validate the effectiveness of step-level state modulation.

2 Related Work

2.1 Methods for Three-Dimensional Bin Packing

Approaches to three-dimensional bin packing include mathematical programming, manually designed heuristics, and learned policies Gzara et al. (2020); Araya & Riff (2014). Exact methods provide optimality guarantees but have limited scalability. Practical solvers more commonly employ constructive rules, local search, or set partitioning Che et al. (2011); Zhu et al. (2012); Santos et al. (2019); Ananno & Ribeiro (2024). These methods typically encode domain knowledge into item-ordering and placement-scoring operators, offering good controllability and ease of deployment. Their performance, however, depends heavily on manually designed decision rules.

Learning-based methods model item selection or placement decisions directly from data Zhang et al. (2021); Que et al. (2023); Jiang et al. (2021). For online and semi-online settings, prior studies address visibility, feasibility, and stability constraints through constrained Markov decision processes Zhao et al. (2023), action-sequence rearrangement Fukuda et al. (2024), or adjustable item ordering Yin et al. (2025); Fang et al. (2025); Pan et al. (2023). These studies highlight the importance of state information but primarily train end-to-end action policies. PackAgent instead evolves executable decision programs offline, preserving explicit geometric checks and information-visibility boundaries.

2.2 LLM-Based Automated Heuristic Design

Automated heuristic design (AHD) aims to reduce manual trial and error in rule construction Van Stein et al. (2024a). Early genetic programming Zhang et al. (2020) and hyper-heuristic Drake et al. (2020) approaches primarily search within spaces of expert-defined primitives. The code-generation capabilities of LLMs further expand the space of candidate programs Van Stein et al. (2024b). For example, FunSearch Romera-Paredes et al. (2024) pairs an LLM with a program evaluator, EoH Liu et al. (2024) co-evolves heuristic ideas and code, EoH-S Liu et al. (2026) searches for complementary heuristic sets, and MCTS-AHD Zheng et al. (2025) uses tree search to balance exploration and exploitation.

Recent studies have also expanded the forms of feedback used in AHD Jiang et al. (2026). ReEvo Ye et al. (2024) uses comparisons between candidates to guide reflective evolution, LLaMEA Van Stein & Bäck (2024) iteratively refines programs based on their execution performance, and teacher-aware evolution extracts behavioral diagnostics from learned policies Chen et al. (2026). PackAgent differs not in its use of performance feedback, but in jointly evolving a state-reflection skill that modulates item-selection and empty maximal space (EMS) scoring factors at every placement step, rather than a fixed ordering or scoring rule.

Most relevant to our work, Sim et al. Sim et al. (2025) show that one-dimensional packing heuristics may lose their advantages across broader instance families. Quan Quan et al. (2025) further demonstrates that optimizing a scoring function within a manually constructed three-dimensional framework is more reliable than generating an entire solver. PackAgent follows this separation of reliable geometric operations from searchable decision modules, but extends the search from a single scoring function to coordinated, state-adaptive skills.

3 Preliminaries

3.1 Problem Definition

Given heterogeneous items $\mathcal{J}$, 3D-BPP seeks a feasible packing into K identical bins while minimizing K. Each bin has dimensions $D_{\text{bin}} = (L, D, H)$, and each item $j \in \mathcal{J}$ has fixed dimensions $d_j = (l_j, w_j, h_j)$. A packing plan specifies each item's bin index k, lower-left corner coordinates (x_j, y_j, z_j), and rotation orientation o_j.

A feasible packing plan must satisfy three constraints: (1) **Containment**: each item lies entirely within the bin dimensions (L, D, H); (2) **Non-overlap**: no two items occupy overlapping volumes; and (3) **Support**: for any item not placed directly on the floor, its contact area with the items below must cover at least 75% of its base area.

3.2 Optimization Objective

We adopt a lexicographic objective: first minimize the number of bins K required to pack all items, and then maximize space utilization Uti among solutions using the same number of bins. The utilization of bin k, where $k \in \{1, \ldots, K\}$, is defined as the ratio of the total item volume in that bin to its capacity:

$$U_k = \frac{\sum_{i=1}^{n_k} l_i w_i h_i}{L \cdot D \cdot H}, \tag{1}$$

where l_i, w_i, and h_i denote the length, width, and height of the i-th item in bin k, respectively, and n_k is the number of items packed into that bin.

In practical warehousing scenarios, the final bin is often only partially filled. To reduce the effect of this residual load, which depends on the total item volume, we define instance-level space utilization as the arithmetic mean of the utilization values of all bins except the last. When only one bin is used, its utilization is used directly:

$$Uti = \begin{cases} \dfrac{1}{K-1}\sum_{k=1}^{K-1} U_k, & K \geq 2, \\ U_1, & K = 1. \end{cases} \tag{2}$$

4 METHOD: PACKAGENT

Figure 1 connects two loops through complete packing outcomes. The inner loop executes an agent's state-adaptive decisions within a fixed multi-pallet greedy solver (Section 4.1); the LLM-driven outer loop evaluates these outcomes and evolves the decision programs through parent selection, evolutionary operators, and elitist updates (Section 4.2).



Figure 1: Overview of PackAgent. Top: LLM-driven three-skill evolution. Bottom: the inner packing loop, where the shared hint from ε modulates ϕ and ψ. Candidate generation α, constraint verification ν, and geometric updates remain fixed.

4.1 PACKING PROCESS

For the item set $\mathcal{J}$ and bin dimensions defined in Section 3, let $\mathcal{R}_t$ denote the remaining items at step t, $\mathcal{B}_t$ the opened pallets and their layouts, and Π_t the completed placement records. Initially, $\mathcal{R}_0 = \mathcal{J}$, $\mathcal{B}_0 = \emptyset$, and $\Pi_0 = \emptyset$. Each step executes

$$\varepsilon \longrightarrow \phi \longrightarrow (\alpha + \nu) \longrightarrow \psi \longrightarrow \text{Placement and Update.} \tag{3}$$

Here, α and ν correspond to candidate generation and feasibility filtering, respectively. Geometric operations and constraint checks are performed by the fixed solver.

4.1.1 PHASE-AWARE META-CONTROL

The eligible item set $\mathcal{E}_t^{\mu}$ is determined by the operating mode $\mu \in \{\text{off}, \text{on}, \text{semi}(\ell)\}$:

$$\mathcal{E}_t^{\mu} = \begin{cases} \mathcal{R}_t, & \mu = \text{off}, \\ \{\text{head}(\mathcal{R}_t)\}, & \mu = \text{on}, \\ \text{first}_{\ell}(\mathcal{R}_t), & \mu = \text{semi}(\ell). \end{cases} \tag{4}$$

The offline mode includes all remaining items, the online mode includes only the item at the head of the queue, and the semi-online mode includes the items within the lookahead window. In the online

and semi-online modes, $\mathcal{R}_t$ preserves the arrival order. The state s_t comprises the current layout, completed placement records, and statistics of visible items. Attributes of items that have not yet arrived or lie outside the lookahead window are excluded from decision-making.

Before each placement, the state reflection skill generates modulation information:

$$h_t = \varepsilon(s_t) = \left(q_t, \mathbf{w}_t^{\phi}, \mathbf{w}_t^{\psi}\right), \tag{5}$$

where q_t is a phase cue, and $\mathbf{w}_t^{\phi}$ and $\mathbf{w}_t^{\psi}$ are the factor weights for item selection and placement scoring, respectively. The evolved program implements the phase identification and weight generation logic. Its outputs are passed to the downstream skills through a shared `phase_hint` interface, allowing both decisions to use the same step-specific modulation information. The skill ε neither directly selects items or placements nor modifies geometric constraints. After each placement, the solver reconstructs the state and generates the next phase hint.

4.1.2 Item Selection

The skill ϕ scores items in the eligible set and selects the next item to place:

$$S_t^{\phi}(j) = \sum_k w_{t,k}^{\phi} f_k^{\phi}(j, s_t), \qquad j_t = \arg\max_{j \in \mathcal{E}_t^{\mu}} S_t^{\phi}(j). \tag{6}$$

The factors f_k^{ϕ} may represent preferences such as volume priority, layer-height compatibility, and shape compactness. Their specific definitions are determined by the agent program, while their weights vary dynamically with the state. In the online mode, the eligible set contains only the head item, which is selected directly without invoking ϕ. State reflection and subsequent placement scoring are still performed at every step.

4.1.3 Candidate Generation and Constraint Verification

The candidate generator α in the fixed solver generates placements for j_t across all opened pallets, enumerating six axis-aligned orientations. The constraint verifier ν checks containment, non-overlap, and at least 75% base-area support. Let P_b denote the current layout of pallet b and Ω the set of six orientations. The feasible candidate set is

$$\mathcal{C}_t = \bigcup_{b \in \mathcal{B}_t} \mathcal{C}_t^b, \tag{7}$$

$$\mathcal{C}_t^b = \{(b, p, o) \mid p \in \alpha(P_b),\ o \in \Omega,\ \nu(j_t, p, o, P_b) = 1\}. \tag{8}$$

All opened pallets remain available for subsequent placements, and feasible candidates are pooled across pallets. A new pallet is opened and candidates are regenerated only when $\mathcal{C}_t$ is empty. If no feasible candidate exists even on an empty pallet, the run is deemed unable to complete the order. Candidate generation, constraint verification, and the pallet-opening policy are controlled by the fixed solver and are unaffected by factor weights.

4.1.4 Placement Scoring and State Update

The skill ψ scores all feasible candidates pooled across pallets:

$$S_t^{\psi}(c) = \sum_r w_{t,r}^{\psi} f_r^{\psi}(c, j_t, s_t), \qquad c_t^* = \arg\max_{c \in \mathcal{C}_t} S_t^{\psi}(c). \tag{9}$$

Placement factors may capture placement height, contact extent, proximity to walls and corners, layer alignment, and compatibility with the remaining space. Their weights are produced by ε at the same step. The fixed solver executes the highest-scoring candidate c_t^*, updates the pallet layout and free space, and removes j_t from the remaining item set. It then reconstructs the state and repeats the process until all items have been placed.

4.2 Evolutionary Process

An evolutionary individual is a complete agent $A = (\varepsilon, \phi, \psi)$. The search space includes both the state-to-weight modulation rules and the downstream decision factors.

4.2.1 POPULATION INITIALIZATION

The system provides the LLM with a task description, function interfaces, and state-visibility constraints, using executable three-skill templates as seeds to generate the initial population. Each offspring is a complete agent, with modifications concentrated on one or two skills and the remaining components inherited from the parent. Modifications must preserve the `phase_hint` communication contract: changes to downstream factors must remain consistent with the weights produced by ε.

Evolution uses EoH-style E1, E2, M1, and M2 operators, supporting exploration of distinct strategies, recombination of effective structures, modification of decision logic, and local parameter adjustment, respectively.

4.2.2 FULL-TRAJECTORY EVALUATION

Current item and placement choices affect subsequent free space and decision conditions. The system therefore executes complete packing trajectories on the same set of training orders and evaluates each agent by its final packing outcomes. Let $\bar{K}(A)$ and $\overline{Uti}(A)$ denote the average pallet count and average space utilization, respectively. The fitness function is defined as

$$F(A) = 1000\,\bar{K}(A) + \left(1 - \overline{Uti}(A)\right). \tag{10}$$

Lower fitness is better. At the training-set size used in this study, this objective prioritizes reducing the average pallet count and then improving utilization, consistent with the lexicographic objective in Section 3.2. Agents with interface errors, execution failures, or invalid actions are marked as invalid. Instances with incomplete packing receive penalties according to the evaluator's rules.

4.2.3 POPULATION UPDATE

At each generation, the parent population and valid offspring are merged, and the best N agents are retained according to fitness. Evolution returns the best agent A^* upon reaching the generation limit or the early-stopping criterion; hyperparameters are reported in Section 5.1.

During deployment, the evolved programs remain unchanged, while ε^* updates hints and weights at each step. No further LLM calls or execution of multiple candidate strategies is required.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

We conduct experiments on the public SSSCSP dataset Kurpel et al. (2020) and the industrial IVC dataset. SSSCSP contains 47 three-dimensional bin packing instances, with the first 20 used for training and the remaining 27 for testing. Item distributions and pallet dimensions vary across instances. The IVC dataset contains item-size distributions from real industrial scenarios and uses a single pallet size, with instances containing 50, 200, or 500 items.

We evaluate performance using the average bin count $\bar{K}$ ($\downarrow$) and space utilization Uti ($\uparrow$), as defined in Equation equation 2. PackAgent uses a population size of $N = 8$ and runs for at most 20 generations, with early stopping after eight consecutive generations without improvement. At each generation, the E1, E2, M1, and M2 operators invoke Qwen3-Max to generate offspring. All variants share the same α and ν functions.

5.2 COMPARATIVE EXPERIMENTS

We compare PackAgent with EoH Liu et al. (2024), ReEvo Ye et al. (2024), and EoH-S Liu et al. (2026) under the offline, online, and semi-online modes defined in Equation equation 4, using lookahead windows $\ell \in \{2, 3, 4\}$ for semi-online packing. Both the online and semi-online modes use three arrival orders: descending base area (`area`), descending height (`height`), and random order.

All comparisons in Table 1 use a single trajectory per instance; for EoH-S, we select a single best rule from its heuristic set.

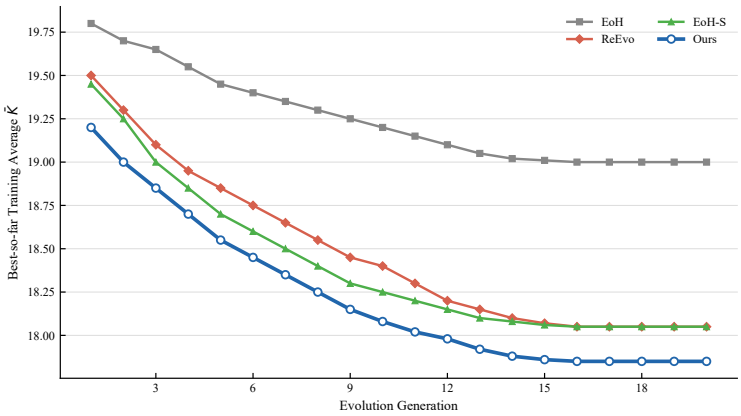


Figure 2: Average bin count $\bar{K}$ on the training set across evolutionary generations for EoH, ReEvo, EoH-S, and PackAgent. Lower values indicate better performance.

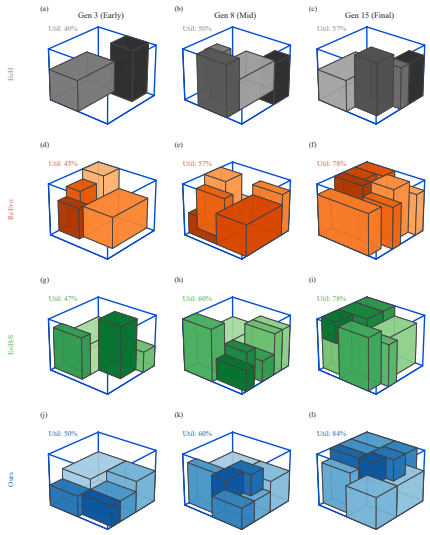


Figure 3: Representative three-dimensional packing layouts produced by EoH, ReEvo, EoH-S, and PackAgent at evolutionary generations $g = 5, 10, 15$.

PackAgent achieves the lowest test average bin count in all three modes, reducing $\bar{K}$ by 0.074, 0.320, and 0.222 bins relative to the best-performing baseline in the offline, online, and semi-online settings, respectively. It also achieves the highest online test utilization of 69.45%. Its offline and semi-online utilization is slightly lower than that of EoH-S and ReEvo, respectively. These results support the effectiveness of state-conditioned adaptation for reducing bin usage within a single trajectory.

Figure 2 shows the average training bin count $\bar{K}$ across evolutionary generations g. All methods gradually converge, while PackAgent consistently maintains a lower $\bar{K}$ and reaches the best final value, consistent with Table 1. Figure 3 presents representative layouts at $g = 5, 10, 15$. The layouts become more compact as evolution progresses, with PackAgent exhibiting more effective space filling.

5.3 Ablation Studies

To directly examine the role of step-level meta-control ε, we fix the best trained three-skill agent $A^* = (\varepsilon^*, \phi^*, \psi^*)$ and intervene only on h_t, the output of Equation equation 5, during deployment. No retraining is performed, and candidate generation, constraint verification, and the mechanism

Table 1: Packing results under offline, online, and semi-online settings. The best value in each column within each setting is highlighted in bold.

Method	Train $\bar{K}$	Test $\bar{K}$	Train Uti (%)	Test Uti (%)
Offline				
EoH	19.000	17.148	63.54	66.24
ReEvo	18.050	15.593	68.35	69.67
EoH-S	18.050	15.222	68.35	**70.75**
PackAgent (Ours)	**17.850**	**15.148**	**69.12**	70.67
Online				
EoH	16.800	15.889	66.52	67.47
ReEvo	16.550	15.629	67.42	68.37
EoH-S	16.625	15.704	66.98	67.93
PackAgent (Ours)	**16.220**	**15.309**	**68.50**	**69.45**
Semi-online				
EoH	17.050	16.062	64.92	65.97
ReEvo	16.550	15.609	**67.73**	**68.68**
EoH-S	16.600	15.683	67.09	68.04
PackAgent (Ours)	**16.320**	**15.387**	67.54	68.49

for filling remaining space on previously opened pallets remain unchanged. We consider four variants. *Empty* ψ and *Empty* ϕ remove the dynamic factor weights of ψ and ϕ, respectively, causing the affected skill to revert to its static default parameters. *Empty reflection* removes the dynamic factor weights of both skills, so that both use their static default weights. *Shuffled hint* breaks the correspondence between h_t and the current state, causing ϕ and ψ to use random weights unrelated to the state.

Table 2 reports the offline results. Full PackAgent performs best on both splits, with test $\bar{K} = 15.148$ and utilization of 70.67%. Removing dynamic weights from either downstream skill degrades performance, with item selection more sensitive than placement scoring. Reverting both skills to static defaults further degrades performance, while *Shuffled hint* performs worst (test $\bar{K} = 16.037$). These results support coordinating both skills through a shared, state-aligned hint rather than merely adding factors or weight parameters.

Table 2: Deployment ablations in the offline setting.

Variant	Train $\bar{K}$	Test $\bar{K}$	Train Uti (%)	Test Uti (%)
Full PackAgent	**17.850**	**15.148**	**69.12**	**70.67**
Empty ψ	18.010	15.519	68.85	69.82
Empty ϕ	18.121	15.852	68.35	69.12
Empty reflection	18.368	15.926	68.05	68.95
Shuffled hint	19.998	16.037	67.45	68.23

5.4 Generalization Experiments

To evaluate the transferability of the evolved rules to instance scales outside the training distribution, we construct three groups of IVC instances containing 50, 200, and 500 items. We compare three methods, each of which must complete an entire offline packing task, including both item selection and placement selection: (1) *Offline*, the original offline packing method Que et al. (2023), which performs both subtasks; (2) *Hybrid*, which combines PackAgent's item selection function ϕ with an online packing method Yin et al. (2025), using the former to select items and the latter to determine their placements; and (3) *PackAgent*, which uses the complete three-skill framework for item selection and placement scoring.

Table 3 shows small differences at 50 items and a growing advantage for PackAgent at larger scales. At 500 items, PackAgent requires 77.8 bins versus 89.6 for Offline. Its utilization remains close to

80% across scales, with reported standard deviation no greater than 1.78, whereas Offline exhibits declining utilization and increasing variability over longer trajectories.

Hybrid reduces the 500-item bin count by 8.2 relative to Offline and improves utilization by 6.3 percentage points, with lower variability. This suggests that the evolved item-selection skill can provide an effective ordering prior for an RL placement policy. Full PackAgent further outperforms Hybrid at every evaluated scale, supporting the value of coordinated item selection and placement scoring.

Table 3: Generalization results across instance scales.

Method	**Items**	$\bar{K}\downarrow$	**Uti (%)**$\uparrow$	**Std.**$\downarrow$
PackAgent	50	**8.1**	**80.2**	**0.82**
	200	**31.2**	**79.5**	**1.61**
	500	**77.8**	**79.3**	**1.78**
Hybrid	50	8.4	78.5	0.91
	200	32.8	77.1	1.85
	500	81.4	76.5	2.05
Offline	50	8.6	77.8	0.95
	200	35.1	73.8	2.34
	500	89.6	70.2	3.12

6 Conclusion

PackAgent enables state-adaptive three-dimensional packing through jointly evolved decision skills. It achieves the lowest average bin count among the compared single-trajectory methods in all three SSSCSP information-visibility settings. IVC experiments support transferability across instance scales, while deployment ablations support the importance of state modulation for coordinating downstream decisions. Future work will evaluate robustness using multiple random seeds and more complex physical constraints.

Reproducibility Statement

We describe the fixed solver, including candidate generation, constraint verification, and geometric updates, in Section 4. The three evolved skills $(\varepsilon, \phi, \psi)$ are implemented as executable program modules with a shared `phase_hint` interface, and their full source code, together with the fixed solver and the evaluation scripts, is provided as anonymized supplementary material. The SSSCSP dataset is publicly available, and we use the first 20 instances for training and the remaining 27 for testing. The construction details of the IVC instances, including item-size distributions and pallet dimensions, are described in the supplementary material. The evolutionary hyperparameters are reported in Section 5.1: population size $N = 8$, at most 20 generations, and early stopping after eight consecutive generations without improvement. Each generation uses the E1, E2, M1, and M2 operators with Qwen3-Max. The fitness function and the evaluation metrics follow Equations equation 10 and equation 2. After evolution, the best agent A^* is frozen; deployment does not call the LLM and only executes the evolved programs. Random seeds and configuration files used in our experiments will be released upon acceptance.

AI Use Statement

This work used a large language model, Qwen3-Max, to assist with implementing methods and designing the research methodology. Specifically, Qwen3-Max was used to generate candidate program modules for state reflection, item selection, and placement scoring, and to guide the evolutionary operators that modify these modules. It was also used to assist with creating software code and editing the manuscript for readability. The model was not used to generate synthetic datasets, to develop theoretical models, or to formulate mathematical claims. All AI-assisted outputs were reviewed and validated by the authors. In particular, every generated skill program was executed

in complete packing simulations on the training orders, and its behavior was checked for geometric feasibility and interface correctness. The authors take full responsibility for the final content, claims, and artifacts of this work. This disclosure is also provided in the OpenReview submission form.

REFERENCES

Anan Ashrabi Ananno and Luis Ribeiro. A multi-heuristic algorithm for multi-container 3-d bin packing problem optimization using real world constraints. *IEEE Access*, 12:42105–42130, 2024.

Ignacio Araya and M-C Riff. A beam search approach to the container loading problem. *Computers & Operations Research*, 43:100–107, 2014.

Chan Hou Che, Weili Huang, Andrew Lim, and Wenbin Zhu. The multiple container loading cost minimization problem. *European Journal of Operational Research*, 214(3):501–511, 2011.

Minyu Chen, Song Qin, Ling-I Wu, Jianxin Xue, and Guoqiang Li. Teacher-aware evolution of heuristic programs from learned optimization policies. *arXiv preprint arXiv:2605.10634*, 2026.

John H Drake, Ahmed Kheiri, Ender Özcan, and Edmund K Burke. Recent advances in selection hyper-heuristics. *European Journal of Operational Research*, 285(2):405–428, 2020.

Han Fang, Paul Weng, and Yutong Ban. Asap: Learning generalizable online bin packing via adaptive selection after pruning. *arXiv e-prints*, pp. arXiv–2501, 2025.

Kenji Fukuda, Masaki Yamakita, and Hiroyuki Oyama. Online 3d bin packing using a combination of tree structures and deep reinforcement learning. In *2024 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 2369–2376. IEEE, 2024.

Ankit Goyal and Jia Deng. Packit: A virtual environment for geometric planning. In *International Conference on Machine Learning*, pp. 3700–3710. PMLR, 2020.

Fatma Gzara, Samir Elhedhli, and Burak C Yildiz. The pallet loading problem: Three-dimensional bin packing with practical constraints. *European Journal of Operational Research*, 287(3):1062–1074, 2020.

Xia Jiang, Yaoxin Wu, Minshuo Li, Zhiguang Cao, and Yingqian Zhang. Large language models as end-to-end combinatorial optimization solvers. *Advances in Neural Information Processing Systems*, 38:164787–164826, 2026.

Yuan Jiang, Zhiguang Cao, and Jie Zhang. Learning to solve 3-d bin packing problem via deep reinforcement learning and constraint programming. *IEEE transactions on cybernetics*, 53(5): 2864–2875, 2021.

Debajyoti Kar, Arindam Khan, and Malin Rau. Improved approximation algorithms for three-dimensional bin packing. *arXiv preprint arXiv:2503.08863*, 2025.

Deidson Vitorio Kurpel, Cassius Tadeu Scarpin, José Eduardo Pécora Junior, Cleder Marcos Schenekemberg, and Leandro C Coelho. The exact solutions of several types of container loading problems. *European Journal of Operational Research*, 284(1):87–107, 2020.

Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. *arXiv preprint arXiv:2401.02051*, 2024.

Fei Liu, Yilu Liu, Qingfu Zhang, Tong Xialiang, and Mingxuan Yuan. Eoh-s: Evolution of heuristic set using llms for automated heuristic design. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 37090–37098, 2026.

Yuxin Pan, Yize Chen, and Fangzhen Lin. Adjustable robust reinforcement learning for online 3d bin packing. *Advances in Neural Information Processing Systems*, 36:51926–51954, 2023.

Julien Pourcel, Cédric Colas, and Pierre-Yves Oudeyer. Self-improving language models for evolutionary program synthesis: A case study on arc-agi. *arXiv preprint arXiv:2507.14172*, 2025.

Guorui Quan, Mingfei Sun, and Manuel López-Ibáñez. Re-evaluating llm-based heuristic search: A case study on the 3d packing problem. *arXiv preprint arXiv:2509.02297*, 2025.

Quanqing Que, Fang Yang, and Defu Zhang. Solving 3d packing problem using transformer network and reinforcement learning. *Expert Systems with Applications*, 214:119153, 2023.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

Luiz FO Moura Santos, Renan Sallai Iwayama, Luísa Brandão Cavalcanti, Leandro Maciel Turi, Fabio Emanuel de Souza Morais, Gabriel Mormilho, and Claudio B Cunha. A variable neighborhood search algorithm for the bin packing problem with compatible categories. *Expert Systems with Applications*, 124:209–225, 2019.

Kevin Sim, Quentin Renau, and Emma Hart. Beyond the hype: Benchmarking llm-evolved heuristics for bin packing. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*, pp. 386–402. Springer, 2025.

N. Van Stein, D. Vermetten, and T. Bäck. In-the-loop hyper-parameter optimization for llm-based automated design of heuristics. *ACM Transactions on Evolutionary Learning*, 2024a.

Niki Van Stein and Thomas Bäck. Llamea: A large language model evolutionary algorithm for automatically generating metaheuristics. *IEEE Transactions on Evolutionary Computation*, 29 (2):331–345, 2024.

Niki Van Stein, Diederick Vermetten, and Thomas Bäck. In-the-loop hyper-parameter optimization for llm-based automated design of heuristics. *ACM Transactions on Evolutionary Learning*, 2024b.

Haoran Ye, Jiarui Wang, Zhiguang Cao, Federico Berto, Chuanbo Hua, Haeyeon Kim, Jinkyoo Park, and Guojie Song. Reevo: Large language models as hyper-heuristics with reflective evolution. *Advances in neural information processing systems*, 37:43571–43608, 2024.

Hao Yin, Chenxi Zhang, Fan Chen, and Hongjie He. Learning three-dimensional bin packing with adjustable-order semi-online setting. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 16088–16094. IEEE, 2025.

Fangfang Zhang, Yi Mei, Su Nguyen, and Mengjie Zhang. Evolving scheduling heuristics via genetic programming with feature selection in dynamic flexible job-shop scheduling. *ieee transactions on cybernetics*, 51(4):1797–1811, 2020.

Jingwei Zhang, Bin Zi, and Xiaoyu Ge. Attend2pack: Bin packing through deep reinforcement learning with attention. *arXiv preprint arXiv:2107.04333*, 2021.

Hang Zhao, Zherong Pan, Yang Yu, and Kai Xu. Learning physically realizable skills for online packing of general 3d shapes. *ACM Transactions on Graphics*, 42(5):1–21, 2023.

Hang Zhao, Juzhan Xu, Kexiong Yu, Ruizhen Hu, Chenyang Zhu, Bo Du, and Kai Xu. Deliberate planning of 3d bin packing on packing configuration trees. *The International Journal of Robotics Research*, 45(8):1195–1216, 2026.

Jie Zhao and Kang Cheong. Structure-aware cooperative ensemble evolutionary optimization on combinatorial problems with multimodal large language models. *Advances in Neural Information Processing Systems*, 38:8730–8758, 2026.

Zhi Zheng, Zhuoliang Xie, Zhenkun Wang, and Bryan Hooi. Monte carlo tree search for comprehensive exploration in llm-based automatic heuristic design. *arXiv preprint arXiv:2501.08603*, 2025.

Wenbin Zhu, Weili Huang, and Andrew Lim. A prototype column generation strategy for the multiple container loading problem. *European Journal of Operational Research*, 223(1):27–39, 2012.